IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of: Theodore F. Emerson et al.	§	Group Art Unit: 2628
Serial No.: 10/611,403	§	Examiner: Nguyen, Hau H.
Filed: July 1, 2003	§	
For: OPERATING SYSTEM INDEPENDENT APPARATUS FOR GRAPHICAL REMOTE ACCESS	§	Atty. Docket:200304331-2 NUHP:0168-1/FLE/PET

Mail Stop Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

CERTIFICATE OF TRANSMISSION OR MAILING
37 C.F.R. 1.8

I hereby certify that this correspondence is being transmitted by facsimile to the United States Patent and Trademark Office in accordance with 37 C.F.R. 1.6(d) or is being deposited with the U.S. Postal Service as First Class Mail with sufficient postage in an envelope addressed to: Mail Stop Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date below:

April 19, 2007 — Melissa Ne
Date — Melissa Neumann

Sir:

DECLARATION OF JEFFERY R. PETERSON

I, Jeffery R. Peterson, hereby declare as follows:

1. I am an associate attorney at the law firm of Fletcher Yoder in Houston, Texas.

2. My business address is set forth below, along with my signature.

3. I worked on the prosecution of the above-identified patent application under the supervision of Mr. Barry D. Blount, who is a shareholder at the law firm of Fletcher Yoder.

4. In working on the prosecution of the above-identified application I sought to contact Wesley Ellinger, a named co-inventor. I was informed that Mr. Ellinger no longer

works at Hewlett-Packard Company (HP). On March 29, 2007, I received Mr. Ellinger's last known contact information from HP and promptly attempted to call him at the phone number provided by HP. After several rings, the phone was answered by an answering machine. The answering machine had a female's voice and did not state as to whom the phone number belonged. I left a message asking Mr. Ellinger to contact me. The phone call was never returned.

5. Additionally, on March 29, 2007, I performed a reverse telephone number search based on the phone number provided by HP. Specifically, I used http://www.anywho.com/qry/wp_rl. A copy of the results of the reverse search is attached herewith as Exhibit C. The reverse search showed that the number no longer belonged to Mr. Ellinger. *See* Exhibit C.

6. Furthermore, on March 30, 2007, I sent a certified letter to Mr. Ellinger at the address provided by HP. The Attached Exhibit D has a copy of the letter as well as a certified mailing receipt of the mailing of the letter. In the letter, I request Mr. Ellinger to call me at my business number so that we could discuss the patent application. *See* Exhibit D. Because Mr. Ellinger had not contacted me in response to the letter, on April 5, 2007, I checked the status of the certified letter. The results of the status check are attached as Exhibit E. As can be seen, the status of the letter is "Undeliverable as Addressed." *See* Exhibit E. I received the unopened certified letter on April 10, 2007. A copy of the returned, unopened letter is attached as Exhibit F. On the returned letter, the address has been crossed-out and an initialed note states, "Attempted unknown vacant apt 6427." Exhibit F, page 1. Additionally, the return receipt is intact. Exhibit F, page 2.

I declare further that all statements made herein are of my own knowledge, are true and that all statements made on information and belief are believed to be true, and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. §1001, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

Dated: April 19, 2007 By: ______________________

Declarant's Full Name: Jeffery R. Peterson

Country of Citizenship: U.S.A.

Business Address: Fletcher Yoder
7915 FM 1960 W
Suite 330
Houston, Texas 77070

EXHIBIT A

Remote Redirection of Graphical Console Data

Version 1.0

REDACTED

Theodore F. Emerson

Corporate Server Design
Industry Standard System Division
Compaq Computer Corp., Houston, TX

REDACTED

Table Of Contents

Table Of Figures

Tables

1 Introduction

The purpose of this document is to address the specific problem of remote console redirection of graphical data. More and more servers are being deployed in environments where the server and the expertise to administer the server are in different physical locations. Additionally, a "lights-out" (no keyboard/monitor) server is highly desirable in many data-centers. In these environments, remote server management and administration is crucial. One key server management technology is the ability to remotely view and control the managed server's console. If this feature is properly implemented, there is little or no need for administrative expertise to be physically present at the server's location.

For such a feature to be useful in various "emergency" situations, it must be as independent as possible from the managed server. Typically, this feature has been implemented in autonomous add-in management cards, such as the Remote Insight Board. This feature has also been embedded directly on the server platform, such as Compaq's Integrated Remote Console. In both of these products, the remote console feature is implemented in such a way as to be entirely independent of software running on the server. This allows the administrator to control the server, regardless of the heath or state of the server. Since the hardware-based remote console function does not require operating system support, the feature can be used to install or configure the operating system as well as when the operating system is off line.

To date, however, such hardware console redirection (or "remote console") has been limited to text video modes only. The primary reasons for this include:

1) ***Complexity*** – The problem of procuring graphical information is very complex mostly due to the nature of the VGA graphics architecture. Since the VGA architecture evolved from several generations of previous graphics controllers, there is a multitude of video modes. These video modes differ greatly in both pixel depth and memory organization. Graphical modes are frequently multi-planar, requiring both I/O and memory operations to access the video frame buffer. Access to graphics video memory frequently can cause side effects and may actually destroy or alter the location being read. Consequently, a management processor cannot reliably read the video frame buffer without disturbing it or interfering with software that is running on the server. The multitude of video modes coupled with the near impossibility of passively reading the video frame buffer have made this feature impossible to implement.
2) ***Bandwidth*** -- Typically, remote management devices communicate through an out-of-band mechanism, such as a serial line or modem. The amount of data required to reproduce the server's screen is staggering, especially in true-color video modes. For instance, if the server video is in a 1024x768 true color display mode, the video frame buffer contains 2,359,296 bytes of relavent information. Transmitting this amount of data through a 28.8kbps connection using compression would take over 8 minutes. This is far too slow for "real-time" updates and is inappropriate for controlling a server in graphics mode.

To address the above problems, graphical remote console programs such as Compaq Carbon Copy and PC/Anywhere have typically been designed to interface directly with the video driver of the operating system. When the operating system wishes to draw a circle, the remote console program intercepts the command and instructs a circle to be drawn on its management console instead of sending the bitmap of the rendered circle. This greatly reduces the bandwidth required to send the graphical data and, since the program intercepts and coordinates with the program rendering the data, there is no need to interrogate the physical frame buffer. These programs work very well as long as the program and operating system are healthy and functioning. Access to graphical video data is not possible before the operating system is loaded or after it has crashed.

The disclosed invention proposes a solution to all of the above problems, providing a method for a management card to procure video information and transmit over a low-bandwidth connection, without coordination or reliance on the operating system. The invention has been prototyped as a firmware upgrade to Integrated Remote Console.

2 Implementation

The invention disclosed provides a solution to the problems mentioned above. The goal is to provide a graphical remote console which is OS and preferably server independent, which is usable under the conditions when a OS-based graphical remote control application would be unavailable. The goal is not to replace such applications, only to provide a means of displaying the graphical context when these applications are unavailable. As a result, the look and feel of this "emergency" remote console does not need to be on par with these applications. The invention has been dubbed "flashlight graphics" which actually is a very good analogy. When the lights are out you would *want* the lights to come back on but you *need* a flashlight. The invention definitely achieves "flashlight" functionality plus a little more.

The preferred embodiment of this invention uses a combination of several different algorithms to achieve the goals mentioned above. The invention uses a modified sampling technique in conjunction with hardware assistance built into certain PCI video graphics controllers to procure data from the graphics frame buffer. The frame buffer is divided *into blocks* which consist of a rectangular block of pixels. The pixels within each block are decimated into grayscale, preferably into 2-bits or 4 gray levels. This decimation greatly reduces the amount of data required to transmit the image, while maintaining readability on the management console. This also essentially collapses all pixel modes into one, meaning that a 24-bpp image takes the same amount of data as a 4-bpp. A hashing algorithm is applied to the decimated pixels within the block to produce a 16-bit hash value. This value can be stored and compared in successive sample periods. If a *block* hash signature differs from a previously stored value, the *block* is transmitted to the management console.

The management processor simply enumerates every *block* in the frame buffer of the managed server applying the above algorithm. When the entire screen is processed, the process is repeated. The result on the management console is a monochrome representation of the managed server's console which is refreshed at a periodic rate. The prototype is capable of one frame every 2 seconds at 640x480 resolution and one frame every 4 seconds at 1024x768. Though somewhat sluggish, this does provide a seemless "flashlight" into the server's console and offers sufficient response time to perform various administrative activities.

A preferred embodiment would use this technique in concert with a software-based solution (Compaq Carbon Copy), switching when appropriate. This would automatically provide the administrator with the "best of both worlds," eliminating the need to switch between utilities when the server's state changes.

The techniques used to implement "flashlight graphics" are disclosed in further detail below.

2.1 Frame Buffer Access

Access to the graphics frame buffer has been a virtual impossibility in traditional VGA designs. (see *complexity* above) Newer PCI VGA controllers, however, allow the video frame buffer to be additionally mapped to a high-address in PCI address space. Certain video controllers, like the ones from ATI Technologies (used in all new server designs) allow deterministic linear access to the video frame buffer from the high-address aperture. Accesses are free from side-effects and require only apron knowledge of the format of the video data. The frame buffer is available on the ATI controllers even in legacy VGA modes. The prototype has been able to successfully interpret the data from all super-VGA and relevant legacy-VGA modes.

2.2 Sampling Techniques

The two techniques commonly used to obtain video data are ***discrete-time sampling*** and ***continuous sampling***. In the ***discrete-time sampling*** approach, the management processor takes a snapshot of video memory at a fixed interval of time. This snapshot is compared to a previous snapshot and preferably the differences are encoded and transmitted. This is the method used on the original ***Server Manager/R*** board and on several competitors' products. This requires access to the video frame buffer and a fairly large buffer to store at least one copy of the video screen. This works for text modes since the data is infrequently modified and there is a relatively small amount of data to collect (~4k). The integrated Remote Console ASIC implements a ***continuous sampling*** technique whereby video data is passively snooped as it is written to the frame buffer and then preferably encoded and transmitted. This technique yields very realistic performance since data is drawn on the management console exactly the same way that it is rendered on the managed server. In graphics modes, however, redundant data is frequently written to the frame buffer. Depending upon the intelligence of the display driver performing the rendering, a ***continuous sampling*** approach would potentially encode a lot of redundant information. Since the data encoded would greatly depend on the activity of the managed server's console and the response time would be bottlenecked by the speed of the transmission medium. The ***discrete-time*** sampling technique, although it can potentially provide a deterministic refresh rate, requires a huge frame buffer (2-4Meg) encode sampled frames.

The "flashlight" approach uses a modified ***discrete-time*** sampling approach. The screen is divided into a plurality of ***blocks***, each containing a rectangular block of pixels. The preferred embodiment uses 16 x 16 pixel ***blocks***, breaking a 1024 x 768 display buffer into 64 x 48 ***blocks***. Each ***block*** is sequentially procured, converted to grayscale, and encoded. Once decimated to a block of gray pixels, a 16-bit hash value is calculated and stored representing the block. In this example, the management processor only need maintain 64 x 48 x 2 bytes/***block***, or 6144 bytes.

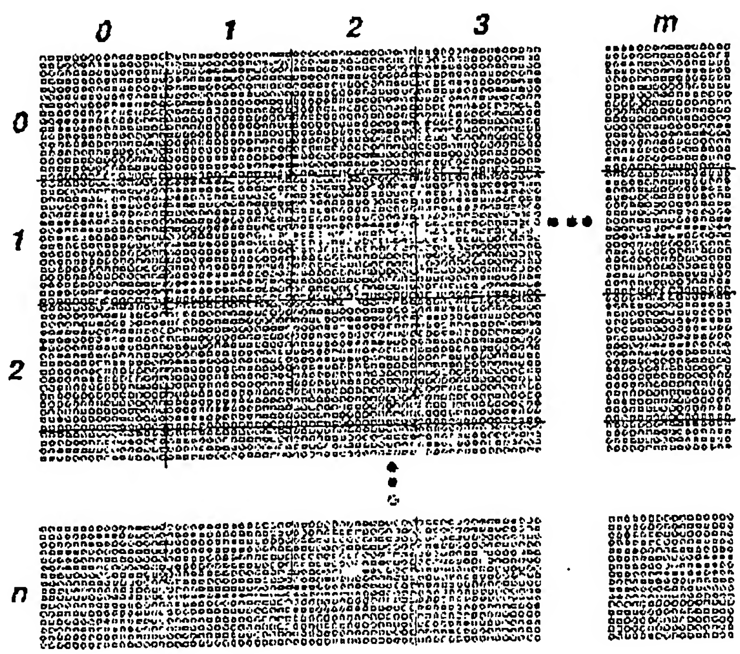

Figure 2-1 – Pixel to Block mapping

Instead of maintaining a complete copy of a previous image sample, the management processor only need keep an array of 16-bit hash entries corresponding to each block.

2.3 Grayscale Conversion

For "emergency" situations, full representation of each color on the managed servers display is unnecessary. Preferably, the encoding algorithm reduces the number of colors to 4 gray shades. This number of shades gives enough color depth to distinguish icons, text, and windows controls on the managed server's screen. This conversion essentially collapses all permutations of color depth into one. That is, once decimated, there is no encoding difference between a 1024x768x4 (16 color mode) and a 1024x768x24 (true color) display mode. This greatly reduces the number of display modes needed to be supported by the management console. Essentially, the management console only needs to know the size of the managed servers display, nothing else.

The standard equation for calculating luminance from a RGB pixel value is:

$$Y=0.3*R+0.59*G+0.11*B$$

Since this formula involves floating point calculations, it is not easily rendered by the management processor. The grayscale conversion preferably uses the following approximation:

$$Y=0.25*R+0.50*G+0.25*B$$

This equation can be calculated very easily with simple register shift instructions.

The resulting image is very readable and more than adequate for "emergency" or OS install applications.

2.4 Hashing Algorithm

The key to enable the management processor to detect modified screen areas is a reliable hash algorithm. From each pixel block, the gray scale values are fed into a hashing algorithm to generate a "signature" for the pixel block. The algorithm preferably uses 16-bit hash values, to reduce the amount of data that must be stored by the management processor. The 16-bit value maps to 65535 different pixel block signatures. Obviously, with only 65535 different possible block signatures, it is possible for two or more different pixel block configurations to map to the same hash value. If this occurs, the management processor would be unable to detect changes to that pixel block. This problem is unavoidable and the hashing algorithm is chosen in such a way as to resist changes that would commonly effect a block. Such operations include, drawing vertical or horizontal lines, etc. Since it is impossible to completely mitigate against hash aliasing, the algorithm will periodically retransmit each block on the screen to insure synchronization. The period for screen synchronization is TBD and may not be necessary for most "emergency" operations that would need to be performed. Another solution is to provide a "refresh" key on the management console when the user detects an area with has fallen out of synchronization due to hash aliasing.

To mitigate hash aliasing, the hashing algorithm generates two independent hash values which are XORed when the entire block has been processed. The 2-bit pixel values are packed into a memory block that is fed to the hashing algorithm.

Hash1=(Hash1+WORD(ptr)) ROL 1
Hash2=Hash2-WORD(ptr)

Once applied to each WORD of the packed pixel block, the two hash values are then XORed to produce a final 16-bit hash signature. The ROL instruction on the first hash function is to mitigate against vertical changes to the pixel block.

Obviously, the hashing algorithm may continuously be refined. The algorithm produces very pleasing results and detects most screen changes very effectively.

2.5 Data Encoding

When a block has been identified for transmission, the decimated grayscale value of the block are transmitted to the management console. This data is usually very compressible. To reduce the amount of data sent to the management console, a data reduction algorithm is performed. Preferably, this is a simple run-length encoding algorithm which is simple to implement on the management processor.

In addition to intra pixel-block run-length encoding, the digital hash signatures can also be used to greatly reduce the number of blocks transmitted to the management console. For instance, if the management console performed a clear screen operation, adjacent pixel blocks would hash to the same value. Upon detecting the same hash value, the management processor might run-length encode pixel blocks – essentially telling the management console to repeat the previous pixel block n times.

Obviously, any data compression technique can be used to reduce the amount of decimated pixel data that needs to be sent to the management console. The communications media may also contain hardware compression algorithms to further increase the data bandwidth available for transmission. (Modem v.34, etc.)

2.6 Detecting Mode Changes

Periodically, the management processor will check the video mode settings of the graphics adapter to insure that the graphics mode has not changed. This is preferably done at the start of each frame. The management processor examines the horizontal and vertical pixel widths as well as the color depth so the graphics frame buffer can be properly interpreted. For indexed color modes, where a palette is used, the palette is interrogated preferably at the start of each frame to detect changes in the graphics color palette. This insures that the data is correctly interpreted and translated to grayscale.

EXHIBIT B

```
        COMPAQ CONFIDENTIAL

        page    58,132
        title   ANSIGRX.ASM
        subttl  Copyright (c) 1982-95 COMPAQ Computer Corporation
;*************************************************************************
;
;       Name:     ANSIGRX.ASM
;
;       Group:  ROM
;
;       Revision:       B.0
;
;       Date:          REDACTED
;
;       Author:        Ted Emerson
;
;*************************************************************************
;
;       Module Functional Description:
;
;       This module contains native->ANSI opcode conversion routines
;
;*************************************************************************
;
;  Changes:
;
;   DATE      REVISION                  DESCRIPTION
;   -----     --------   ------------------------------------------------------
;  REDACTED     0.0      Technology investigation
;*************************************************************************

;*******************************************************************
;       Segment and extrns
;*******************************************************************
```

REDACTED

REDACTED

REDACTED

REDACTED

REDACTED

REDACTED

REDACTED

```
HashLoop:
        mov     eax,dword ptr cs:[si+offset BlockBuf]    ;Grab next Line
        add     si,4
        add     dx,ax
        sub     bx,ax
        rol     dx,1
        shr     eax,16
        add     dx,ax
        sub     bx,ax
        rol     dx,1
        loop    HashLoop
        xor     dx,bx
;DX now contains the hash for this block.  Compare it to the hash table
        mov     bx,word ptr cs:[nBlockCount]
        shl     bx,1                    ;Get word address
        cmp     dx,word ptr [bx+wVidMem]        ;Is the hash value the same?
        jnz     short BlockDiff

;Ok, the block has not been modified.
BlockUnmodified:
        test    byte ptr ds:[GRX_STATUS],fGrxAnchor
;       cmp     byte ptr cs:[fAnchor],0 ;Are we anchored?
        jz      short BlockUnmodUnanchored      ;Are we anchored?

        call    FlushRLE                        ;Flush out old stuff
;       mov     byte ptr cs:[fAnchor],0 ;Raise the anchor
        and     byte ptr ds:[GRX_STATUS],NOT fGrxAnchor ;Raise the anchor
BlockUnmodUnanchored:
        ret

BlockDiff:
        mov     word ptr [bx+wVidMem],dx                ;Update hash table
;       cmp     byte ptr cs:[fAnchor],1         ;Are we anchored?
;       jz      short BlockDiffAnchored
        test    byte ptr ds:[GRX_STATUS],fGrxAnchor     ;Are we anchored?
        jnz     short BlockDiffAnchored
;Ok, blocks has been modified and we are not currently anchored.  Send out new address
; and set anchor.
;       mov     byte ptr cs:[fAnchor],1         ;Drop anchor
        or      byte ptr ds:[GRX_STATUS],fGrxAnchor     ;Drop anchor

        mov     al,GRXESCOPCODE         ;Send out Escape code
        call    EmitByteRaw
        mov     al,GRXSETCURSOR         ;Send out Set cursor opcode
        call    EmitByteRaw
        mov     al,byte ptr cs:[nCol]
        call    EmitByteRaw
        mov     al,byte ptr cs:[nRow]
        call    EmitByteRaw

BlockDiffAnchored:
```

Section C

Section D

Section E

```
        mov     si,0
        mov     cx,GRXBLOCKHEIGHT*4
BlockDiffSendLp:
        mov     al,cs:[si+offset BlockBuf]
        add     si,1
        call    EncodeByte
        loop    BlockDiffSendLp
        ret
```

Section E

```
; Encode 8-bits per pixel video data
;   ESI points to video frame buffer
;
;   Uses:
;       CL pixel count horizontal
;       CH pixel count vertical
;       EDX pixel accumulator
;       BX index register to point into palette buffer
Encode8:
        push    esi                     ;Save registers
        push    es
        mov     ax,VIDSEL
        mov     es,ax           ;Load selector of memory base (SMI will be 0)
        mov     cx,0
        mov     bx,0
        xor     edx,edx

Encode8LineLp:
        mov     eax,es:[esi]            ;Grab 8-bit pixel value
        add     esi,4                   ;Move to next pixel

Encode8SetLp:
        mov     bl,al
        shr     eax,8
        mov     bl,cs:[nPalBuf+bx]      ;Load decimated value
        rol     edx,GRXGRAYDEPTH        ;Shift up
        or      dl,bl                   ;Place in pixel accumulator

        inc     cx                      ;Increment pixel number
        test    cl,00000011b    ;Are we done with this set of pixels?
        jnz     short Encode8SetLp

        test    cl,(32/GRXGRAYDEPTH)-1  ;Packing 32/GRXGRAYDEPTH bits into DWORD
        jnz     short Encode8LineLp

        mov     eax,edx
        call    SaveGrxDWORD            ;Send this GRX data
        xor     edx,edx                 ;Clear out pixel accumulator
;       test    cl,GRXBLOCKWIDTH-1      ;These test are not needed if blockwidth = 16
;       jnz     short Encode8LineLp       ;These test are not needed if blockwidth = 16
;We have encoded a line of GRXBLOCKWIDTH pixels
        add     esi,dword ptr cs:[dwNextLine]     ;Move index register to next line in block
        test    cx,(GRXBLOCKWIDTH*GRXBLOCKHEIGHT)-1     ;Are we done with this block?
        jnz     short Encode8LineLp

        pop     es
        pop     esi                     ;Restore registers
        ret
```

Section B

REDACTED

REDACTED

REDACTED

REDACTED

REDACTED

REDACTED

```
GetModeInfo:
        push    es
        mov     ax,VIDSEL
        mov     es,ax               ;Load selector of memory base (SMI will be 0)

        and     byte ptr ds:[GRX_STATUS],NOT fResChange ;Assume no resolution change
        mov     eax,dword ptr cs:[nbScreenWidth]
        mov     dword ptr cs:[nbOldScreenWidth],eax     ;Store away old mode

        xor     eax,eax
        mov     dword ptr cs:[dwVTOP],eax
ifdef IRC
        mov     ebx,dword ptr cs:[dwAperatureBaseAdr]
        mov     ax,word ptr es:[ebx+7ffc00h+CRTC_H_DISP]    ;Get horizontal display end (pixels*8)
        xor     ah,ah               ;Upper 8-bits of this register are reserved
        inc     ax                          ;Value is 1 less than you would expect
        shr     ax,1                ;Value is pixels*8, divide by 2 to get number of blocks (16 pixels
wide)
else
        mov     ax,320/16
endif
        mov     cs:[nbScreenWidth],ax
ifdef PARANOID
        mov     dx,10f0h
        out     dx,ax
endif

ifdef IRC
        mov     ax,word ptr es:[ebx+7ffc00h+CRTC_V_DISP]    ;Get horizontal display end (pixels*8)
        and     ax,07ffh                    ;Upper bits reserved
        inc     ax
        shr     ax,4                ;Value is in lines.  Divide by 16 to get number of blocks (16 lines
high)
else
        mov     ax,200/16
endif
        mov     cs:[nbScreenHeight],ax
ifdef PARANOID
        mov     dx,10f0h
        out     dx,ax
endif
```

Section A

REDACTED

REDACTED

REDACTED

REDACTED

REDACTED

REDACTED

```
EncodeNOP:
        ret

        align   4
dwEncodeTable   dw      offset EncodeMode13
                dw      offset Encode4
                dw      offset Encode8
                dw      offset EncodeNOP
                dw      offset Encode16
                dw      offset Encode24
```

REDACTED

REDACTED

REDACTED

EXHIBIT C

Google WEB SEARCH GO

FIND A BUSINESS OR PERSON BY PHONE NUMBER

Area Code *Required* 281
Telephone Number *Required* 8071884
SEARCH

TIP: Cell phone numbers are not available

You searched for: 281 8071884
Results 1 - 1 of 1

PREVIOUS | NEXT

Reverse Telephone Listings

Alexander, M T **281-807-1884**
7979 N Eldridge Pkwy
Houston, TX 77041

Maps & Directions | Did you go to school with M T Alexander?
Find All M T Alexander's Info Here!
Instant Background Check Available - $49.95!
Find a Nearby Business

PREVIOUS | NEXT

Reverse Lookup
No Results?
Try Again Here

xxx xxxxxxx

Search

Useful Links

- Find People
- Satellite Photo
- Unlisted Numbers
- Address Finder
- Criminal Records
- Locate Anyone
- Public Records
- Background Searcl
- Reverse Lookup
- More Info

EXHIBIT D

FLETCHER YODER

A Professional Corporation

Attorneys at Law

7915 FM 1960 West, Suite 330
Houston, Texas 77070

Post Office Box 692289
Houston, Texas 77269-2289

Telephone (281) 970-4545
Facsimile (281) 970-4503

200304331-2
NUHP:0168-1

March 30, 2007

Wesley Ellinger
10910 Gold Pt #1904
Houston, TX 77064
Phone: 281-807-1884

Dear Mr. Ellinger:

I am currently working on a patent application assigned to your former employer, Hewlett-Packard Company, on which you are named as a co-inventor. Please contact me as soon as possible at (281) 970-4545 so that we may discuss the application.

I look forward to hearing from you soon.

Regards,

Jeffery R. Peterson

Jeffery R. Peterson

cc: Michael G. Fletcher, Esq.
Barry D. Blount, Esq.

SENDER: COMPLETE THIS SECTION

- Complete items 1, 2, and 3. Also complete item 4 if Restricted Delivery is desired.
- Print your name and address on the reverse so that we can return the card to you.
- Attach this card to the back of the mailpiece, or on the front if space permits.

1. Article Addressed to:

Wesley Ellinger
10910 Gold Pt. #1904
Houston, TX 77064

NUHP: 0168-1

COMPLETE THIS SECTION ON DELIVERY

A. Signature
X ☐ Agent ☐ Addressee

B. Received by (Printed Name) | C. Date of Delivery

D. Is delivery address different from item 1? ☐ Yes
If YES, enter delivery address below: ☐ No

3. Service Type
☒ Certified Mail ☐ Express Mail
☐ Registered ☐ Return Receipt for Merchandise
☐ Insured Mail ☐ C.O.D.

4. Restricted Delivery? (Extra Fee) ☐ Yes

2. Article Number
(Transfer from service label) 7002 0860 0006 3191 6511

PS Form 3811, August 2001 Domestic Return Receipt 102595-02-M-1035

PLACE STICKER AT TOP OF ENVELOPE TO THE RIGHT OF THE RETURN ADDRESS, FOLD AT DOTTED LINE

CERTIFIED MAIL

7002 0860 0006 3191 6511

7002 0860 0006 3191 6511

U.S. Postal Service
CERTIFIED MAIL RECEIPT
(Domestic Mail Only; No Insurance Coverage Provided)

OFFICIAL USE

Postage	$ 0.63
Certified Fee	2.40
Return Receipt Fee (Endorsement Required)	1.85
Restricted Delivery Fee (Endorsement Required)	
Total Postage & Fees	$ 3.88

Postmark Here

PET/mn

Sent To: Wesley Ellinger

Street, Apt. No.; or PO Box No.: 10910 Gold Pt. #1904

City, State, ZIP+4: Houston TX 77064

PS Form 3800, April 2002 See Reverse for Instructions

EXHIBIT E

UNITED STATES POSTAL SERVICE®

Home | Help | Sign In

Track & Confirm FAQs

Track & Confirm

Search Results

Label/Receipt Number: **7002 0860 0006 3191 6511**
Status: **Undeliverable as Addressed**

Your item was undeliverable as addressed at 11:08 AM on March 31, 2007 in HOUSTON, TX 77064. It is being returned if appropriate information is available.

Additional Details > Return to USPS.com Home >

Track & Confirm

Enter Label/Receipt Number.

Go >

Notification Options

Track & Confirm by email

Get current event information or updates for your item sent to you or others by email. Go >

POSTAL INSPECTORS
Preserving the Trust

site map contact us government services jobs National & Premier Accounts

 Terms of Use Privacy Policy

EXHIBIT F

Fletcher Yoder
P.O. Box 692289
Houston, TX 77269-2289

UNITED STATES POSTAL SERVICE
0000
77064
U.S. POSTAGE
PAID
HOUSTON, TX
77070
MAR 30, 07
AMOUNT
$0.76
00016696-04

CERTIFIED MAIL

7002 0860 0006 3191 6511

USA $3.85

3c USPS USA HOUSTON TEXAS

~~Wesley Ellinger~~
~~10910 Gold Pt #1904~~
~~Houston, TX 77064~~

ATTEMPTED
UNKNOWN
VACANT
APT
6/27

SENDER: *COMPLETE THIS SECTION*

- Complete items 1, 2, and 3. Also complete item 4 if Restricted Delivery is desired.
- Print your name and address on the reverse so that we can return the card to you.
- Attach this card to the back of the mailpiece, or on the front if space permits.

1. Article Addressed to:

Wesley Ellinger
10910 Gold Pt. #1904
Houston, TX 77064

NUHP: 0168-1

COMPLETE THIS SECTION ON DELIVERY

A. Signature
X ☐ Agent ☐ Addressee

B. Received by (*Printed Name*) | C. Date of Delivery

D. Is delivery address different from item 1? ☐ Yes
If YES, enter delivery address below: ☐ No

3. Service Type
☒ Certified Mail ☐ Express Mail
☐ Registered ☐ Return Receipt for Merchandise
☐ Insured Mail ☐ C.O.D.

4. Restricted Delivery? (*Extra Fee*) ☐ Yes

2. Article Number
(*Transfer from service label*) 7002 0860 0006 3191 6511

PS Form 3811, August 2001 Domestic Return Receipt 102595-02-M-1035